Implicit and Explicit Graph Embedding: Comparison of both Approaches on Chemoinformatics Applications

Benoit Gaüzère, Hasegawa Makoto, Luc Brun, Salvatore Tabbone

Implicit and Explicit Graph Embedding: Comparison of both Approaches on Chemoinformatics Applications.

Benoit Gaüzère[1], Hasegawa Makoto[2], Luc Brun[1], and Salvatore Tabbone[2]

[1] Université de Caen - Basse Normandie-GREYC CNRS UMR 6072, Caen, France,
[2] Université de Lorraine-LORIA CNRS UMR 7503, Nancy, France

Abstract. Defining similarities or distances between graphs is one of the bases of the structural pattern recognition field. An important trend within this field consists in going beyond the simple formulation of similarity measures by studying properties of graph's spaces induced by such distance or similarity measures . Such a problematic is closely related to the graph embedding problem. In this article, we investigate two types of similarity measures. The first one is based on the notion of graph edit distance which aims to catch a global dissimilarity between graphs. The second family is based on comparisons of bags of patterns extracted from graphs to be compared. Both approaches are detailed and their performances are evaluated on different chemoinformatics problems.

1 Introduction

Graphs allow to encode not only the elementary features of a set of objects but also the relationships between these objects. Graphs constitute thus an efficient tool to model complex objects or phenomenons. Classification, regression or clustering operations applied on graphs constitute an important sub field of the structural pattern recognition framework, all these operations being based either implicitly or explicitly on a distance or a similarity measure.

Definition of graph distances or graph similarity measures constitute an active field within the structural pattern recognition framework. Main distance definitions are based on one hand on the size of the minimum common super graph or the maximum common sub graph and on the other hand on the minimal number of vertex/edge insertion/removal/relabeling required to transform one graph into an other. This last measure called the edit distance is related to the notion of maximum common sub graph [2] and provides a nicely interpretable measure of distance between two graphs. Moreover, assuming basic properties on the elementary edit costs, one can show that this distance satisfies the 4 properties of a distance (positivity, separation, symmetry and triangular inequality). However, the number of calculus required by edit distance computation grows exponentially with the number of nodes of both input graphs and several heuristics have been proposed to obtain efficient but sub optimal edit distances [8,14].

Restricting structural pattern recognition to pairwise comparisons of graphs leads to restrict the field to efficient but often basic classification or clustering algorithms such as the k-nearest neighbor or the k-median algorithms. Computing efficiently more global feature on a set of graphs requires additional properties of the topology of graph's space implicitly defined by a distance measure between graphs. Such a problem may be solved by defining a natural embedding of graphs. Such an embedding leads to associate explicitly or implicitly a vector to each graph and to define a metric between these vectors which corresponds to the metric defined by the graph distance. However, the fact that a distance satisfies the 4 usual distance's axioms does not insure that an embedding within an Hilbert space may be associated to graphs [3]. More precisely, given a set of n graphs, and a matrix D encoding all pairwise distances between the graphs of the set, the type of space induced by D is provided by the spectrum of the matrix $S^c = -\frac{1}{2}(I - \frac{1}{n}ee^t)D(I - \frac{1}{n}ee^t)$ where e is the vector of ones (Section 2). The metric space encoding similarities between graphs is a Krein space if this spectrum contains negative eigen values and an Hilbert space otherwise. Krein spaces have unusual properties such as possibly negative distances between graphs. In order to avoid to use such spaces, several authors [8] regularize the matrix S^c in order to remove its negative eigen values hereby slightly modifying the original metric defined by D. An alternative approach consists in associating a vector to each graph using for example spectral analysis [12]. The approach is in this case slightly different since the metric defined between vectors does not correspond to a metric initially defined in the graph space. A last approach consists in defining a symmetric similarity measure between graphs. The matrix encoding all pairwise similarities between the graphs of a set is called the Gram matrix of this set. If for some sets of graphs the Gram matrix is non definite positive the embedding space associated to this similarity measure is a Krein space. Otherwise, the embedding space corresponds to an Hilbert space and the similarity measure is called a kernel. In this last case the similarity function corresponds to a scalar product between the vectors associated to both input graphs. One may note the symmetry between embeddings based on distances and similarity measures. Both problems are indeed related, since within an Hibert space or a Krein space a distance measure may be defined from scalar products and conversely.

This paper provides a comparison of both distance and similarity approaches. We first present two important methods within the distance based embedding framework in Section 2. Then we provide an overview of graph kernels methods in Section 3. Both approaches are finally compared in Section 4 on several chemoinformatics data sets.

2 Graph embedding

Embedding graph in vector space aims to define points in a vector space such that their mutual distances is as close as possible to the initial graph dissimilarity matrix wrt a cost function (eg. graph edit distance). More precisely, let $G=\{g_1, ..., g_n\}$ be a set of graphs and d: $G \times G \rightarrow \mathbb{R}$ a graph distance function

between pairs of its elements and let $D = D_{ij} = d(g_i, g_j) \in \mathbb{R}^{n \times n}$ be the dissimilarity matrix. The aim in the graph embedding is to provide n p-dimensional vectors x_i such that the distance between x_i and x_j is close as possible to the similarity D_{ij} between g_i and g_j. Thus, embedding graph into a vector space make the graph available to numerous machine learning techniques which require vectorial representation.

Numerous approaches [4,8,12,18] have been proposed in the literature. In this paper we recall the approach proposed in [8] and which is based on the constant shift embedding [15]. Originally, the constant shift embedding was introduced in order to embed pairwise data into Euclidean vector spaces. In [8], the authors adapt this method to the domain of graphs. The key issue is to convert general dissimilarity data into metric data.

Constant shift embedding. We briefly describe the method of Roth et al. [15] to embed D (restricted by the constraint that self-dissimilarities are equal to zero) into a Euclidian space, without influencing the distribution of the initial data. The aim of this approach is to determine a matrix $\tilde{D}$ close to D such that it exists a set of vectors $(x_i)_{i \in \{1,\ldots,n\}}$ with $\tilde{D}_{ij} = \|x_i - x_j\|^2$. The solution of this problem is of course not unique since any translation of vectors x_i would provide a same distance matrix. In order to overcome this problem we perform a centralization of matrix D by considering $S^c = -\frac{1}{2} D^c$, where $D^c = QDQ$ is the definition of the centralization and $Q = I_n - \frac{1}{n} e_n e_n^\intercal$ is the projection matrix on the orthogonal complement of $e_n = (1, \ldots, 1)$. Such a matrix S^c satifies:

$$D^c_{ij} = S^c_{ii} + S^c_{jj} - 2S^c_{ij} \tag{1}$$

If S^c is semidefinite positive, its singular value decomposition is equal to $S^c = V \Lambda V^t$ where columns of V encode the eigen vectors of S^c and Λ is a diagonal matrix encoding its positive eigen values. Setting $X = V\,(\Lambda)^{\frac{1}{2}}$, we obtain $S^c = XX^t$. Hence, each element S^c_{ij} of S^c is equal to a scalar product $< x_i, x_j >$ between the lines i and j of X. Equation 1 may thus be intepreted as a classical result on Euclidean norms stating that the squared distance between two vectors is equal to the sum of the squared norms of these vectors minus twice their scalar product. The scaled eigen vectors $(x_i)_{i \in \{1,\ldots,n\}}$ provide thus a natural embedding of matrix D when matrix S^c is definite positive.

Following the constant shift embedding S^c can be transformed into a positive semidefinite matrix (see Lemma 2 in [15]):

$$\tilde{\mathrm{S}} = S^c - \lambda_n(S^c) I_n$$

where $\lambda_n(S^c)$ is the minimal eigenvalue of the matrix S^c. The diagonal shift of the matrix S^c transforms the dissimilarity matrix D in a matrix representing squared Euclidean distances. The resulting embedding of D is defined by (minimal shift theorem):

$$\tilde{\mathrm{D}}_{ij} = \tilde{\mathrm{S}}_{ii} + \tilde{\mathrm{S}}_{jj} - 2\tilde{\mathrm{S}}_{ij} \Longleftrightarrow \tilde{\mathrm{D}} = D - 2\lambda_n(S^c)(e_n e_n^\intercal - I_n)$$

Setting dimension In PCA it is known that small eigenvalues contain the noise. Therefore, the dimensionality p can reduced by choosing $t \leq p$. Consequently, a $n \times t$ map matrix $X_t{=}V_t(\Lambda_t)^{1/2}$ will be computed where V_t is the column-matrix of the selected eigenvectors and Λ_t the diagonal matrix of the corresponding eigenvectors.

Graph similarity measure Let us recall how the similarity (or dissimilarity) in the domain of graphs can be computed. Similarity between two graphs is almost always referred as a graph matching problem. Graph matching is the process of finding a correspondence between nodes and edges of two graphs that satisfies some constraints ensuring that similar substructures in one graph are mapped to similar substructures in the other. Many approaches have been proposed to solve the graph matching problem. Among these, the graph edit distance has been widely used as the most appropriate similarity measure for representing the distance between graphs. In this paper we use two approaches [7, 14] based both on an approximation of the graph edit distance as an instance of an assignment problem where the edit distance between two graphs is based on a bipartite graph matching. In both approaches, the authors formulate the assignment problem by cost matrix where the optimal match is solved by the Hungarian algorithm.

In [14], each entry of the cost matrix encodes the cost of a node substitution, deletion or insertion. Substitution costs are defined using the Hungarian algorithms on the set of incident edges of both vertices. The substitution cost of two incident edges takes into account the label of the edge and the label of its incident vertices.

In [7] the cost matrix is encoded differently on a distance (HEOM distance) of node signatures. A signature describes the node (degree, attributes), the incident edges attributes but also the degrees of the adjacent nodes. The main differences with the previous approach is that no prior computation (learning phase) of the edit cost function are needed and more global information are taken into account on the graph in the signature.

3 Graph Kernels Methods

Graph embedding methods aim to associate coordinates to graphs. Such an embedding allows us to define similarity or distance measures from graph's coordinates. An alternative strategy consists in computing directly a similarity measure between graphs. Graph kernels can be understood as symmetric graph similarity measures. Using a semi definite positive kernel, the value $K(G, G')$, where G and G' encode two input graphs corresponds to a scalar product between two vectors $\phi(G)$ and $\phi(G')$ in some Hilbert space, called feature space. Distance between two graphs G and G' can be retrieved from kernel function by the relation (Equation 1) $d^2(G, G') = K(G, G) + K(G', G') - 2K(G, G')$. Thanks to this possibly implicit embedding of graphs into an Hilbert space, graph kernels can be combined with machine learning methods based on scalar products between input data, such as the well-known SVM. This use of kernels

into statistical machine learning method, called kernel trick, provides a natural connection between structural pattern recognition and graph theory on one hand and statistical pattern recognition on the other hand.

A large family of graph kernels are based on the extraction of a bag of patterns from each graph. Methods corresponding to this family consists in three key steps. First, bags of pattern are built from graphs by enumerating a given set of patterns $\mathcal{P}$ within graphs. This enumeration, possibly implicit, defines an embedding of graphs into a feature space where each dimension is associated to a pattern. Second, global similarity between graphs is defined by the similarity of their bags of patterns. Finally, this similarity between bags is based on a sub kernel between pattern $k_p : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$. This sub kernel k_p encodes the similarity of two patterns extracted from graphs.

A common approach defines the set of patterns as all possible walks included within a graph. A first method, defined by Gärtner and al., proposes a formulation of a kernel based on graph product and powers of adjacency matrix [5] which computes the number of common walks of the two graphs to be compared. A second method proposed by Kashima and al. [9] defines a random walk kernel by considering the probability $p(w|G)$ of encountering a random walk w within a graph G. Using such probabilities, the kernel is defined as:

$$k_{rw}(G, G') = \sum_{w \in \mathcal{W}(G)} \sum_{w' \in \mathcal{W}(G')} p(w|G)p(w'|G')k(w, w') \tag{2}$$

with $\mathcal{W}(G)$ denoting the set of walks extracted from G. Vishwanathan [16] has proposed an unified and efficient computation of both methods by means of Sylvester equations. However, comparison of graphs based on random walks suffers from tottering. Tottering corresponds to possible infinite oscillations between two nodes which leads to artificially long walks not representative of the structure of the graphs.

The major drawback of methods based on linear patterns is that linear structures can not represent most of the structural information encoded within complex and non linear structures such as molecular graphs. In order to tackle this limitation, Ramon and Gärtner [11] and Mahé and Vert [10] have proposed a kernel based on the comparison of non linear patterns. This set of non linear patterns is defined as the set of tree patterns, denoted $\mathcal{T}_P$, i.e. trees where a same node can appears more than once. This kernel maps each tree pattern having a different labeling to a specific dimension in an infinite feature space representing all possible tree patterns. This embedding may be encoded by projection $\phi_{\mathcal{T}_P}(G)$ and graph kernel is defined as an inner product between these projections: $K_{\mathcal{T}_P}(G, G') = \langle \phi_{\mathcal{T}_P}(G), \phi_{\mathcal{T}_P}(G') \rangle$. Computation of this kernel is based on a recursive comparison of neighborhood matching sets up to a given depth [10].

Mahé and Vert have proposed in [10] an extension of tree pattern kernel which weights each tree pattern according to its structural complexity. This measure of structural complexity may be encoded by the branching cardinality or the ratio between number of nodes and depth of tree patterns. However, since the number of occurrences of each tree pattern is not explicitly computed during

kernel computation, only an a priori weighting of tree patterns can be applied to each tree pattern. In addition, as observed on walks, tree patterns suffers from tottering. However, Mahé and Vert [10] have proposed an extension to prevent tottering which consists in transforming input graphs.

Another method based on non linear patterns computes an explicit distribution of each pattern within a graph. This method, called treelet kernel [6], explicitly enumerates treelets included within a graph, the set of treelets being defined as the 14 trees having a size lower than or equals to 6 nodes. Thanks to the limited number of different patterns encoding treelets, an efficient algorithm allows to enumerate the number of occurrences of each pattern within a graph. Given this first enumeration, a first kernel on unlabeled graphs can be defined. When applying this method to set of labeled graphs, labeling information included within treelets is encoded by a canonical key. This canonical key is defined such as if treelets have a same structure, their canonical key is similar if and only if the two treelets are isomorphic. Each treelet being uniquely identified by the index of its pattern and its canonical key, any graph G can be associated to a vector $f(G)$ which explicitly encodes the number of occurrences of each treelet t by $f_t(G)$. Using this vector representation, treelet kernel between graphs is defined as a sum of sub kernels between common treelets of both graphs:

$$K_{\mathcal{T}}(G, G') = \sum_{t \in \mathcal{T}(G) \cap \mathcal{T}(G')} k(f_t(G), f_t(G')) \tag{3}$$

where $k(.,.)$ defines any positive definite kernel between real numbers such as linear, Gaussian or polynomial kernel. In the same way as tree pattern kernel, each pattern can be weighted in order to improve kernel accuracy as follows:

$$K_{\mathcal{T}}(G, G') = \sum_{t \in \mathcal{T}(G) \cap \mathcal{T}(G')} w(t) k(f_t(G), f_t(G')) \tag{4}$$

However, conversely to tree pattern kernel, the explicit enumeration of each sub structure provided by treelet kernel method allows to weight each pattern according to a property to predict and not only according to an a priori function. This weighting may be computed using variable selection algorithms [6] or multiple kernel learning [1].

4 Experiments

Our first experiment is based on two regression problems[3] which consist in predicting molecule boiling points. The first dataset is composed of 150 alkanes, an alkane corresponding to an acyclic molecule solely composed of carbons and hydrogens. A common encoding is to implicitly encode hydrogen atoms using the valency of carbon atoms . Such an encoding scheme allows to represent alkanes

[3] These databases are available on the IAPR TC15 Web page: http://www.greyc.ensicaen.fr/iapr-tc15/links.html#chemistry

Table 1. Boiling point prediction.

Method	RMSE (°C)		Computation
	Alkane	Acyclic	Time (s)
(1) Gaussian edit distance	10.01	10.27	1.35
(2) Random Walks Kernel	16.28	18.72	19.10
(3) Tree Pattern Kernel	3.48	11.02	4.98
(4) Treelet Kernel	**1.92**	**8.10**	**0.07**
(5) Graph Embedding	6.15	12.3	7.21

as unlabeled graphs. The second dataset is composed of 183 acyclic molecules, each molecule being composed of heteroatoms and thus encoded as acyclic labeled graphs. We evaluate the boiling point of each molecule using several test sets composed of 10% of the database, the remaining 90% being used as training set. First, we can note that linear patterns (Table 1, Line 2) do not encode enough structural information to correctly predict boiling points of molecules. Conversely, methods based on bags of non linear patterns obtain better results (Table 1, Lines 3 and 4). Differences between Treelet Kernel and Tree Pattern Kernel may be explained by the use of a Gaussian kernel for Treelet kernel, which is not possible with tree pattern computation scheme. In addition, limitation on the size of patterns induced by explicit enumeration of treelets does not have a lot of influence on these problems since molecules have a low number of atoms. Second, Table 1 show results obtain by graph embedding method (Line 5) and a Gaussian kernel applied on the approximate edit distance as defined by [14] (Line 1). Graph embedding results have been computed using different subsets of eigenvalues obtained by applying a threshold on variance encoded within the matrix.We can note that the improvement on edit distance approximation leads to better results than approximation defined in [14] when applied to unlabeled graphs. Finally, the last column of Table 1 shows the time required to compute the Gram matrix on acyclic dataset. Since most of the methods are computed within the same order of magnitude (seconds), Treelet Kernel can be computed in 0.07 seconds thanks to the efficient enumeration of a limited set of patterns.

The second experiment consists of two classification problems. The first one is taken from the Predictive Toxicity Challenge [17] which aims to predict carcinogenicity of 416 chemical compounds applied to female (F) and male (M) rats (R) and mice (M). This experiment consists of ten different datasets for each class of animal, each of them being composed of one train set of about 310 molecules and one test set of about 35 molecules. The second dataset is provided by [13]. This database defined from the AIDS Antiviral Screen Database of Active Compounds is composed of 2000 chemical compounds. These chemical compounds have been screened as active or inactive against HIV and they are split into three different sets. A train set composed of 250 compounds used to train SVM, a validation set composed of 250 compounds used to find parameters giving the best prediction accuracy and a test set composed of remaining 1500 compounds. Table 2 shows the amount of correctly classified molecules over

Table 2. Classification accuracy on the two classification experiments.

Method	PTC				AIDS
	MM	FM	MR	FR	
(1) Gaussian Edit Distance	**223**	212	194	234	**99.7%**
(2) Random Walks Kernel	216	221	201	232	98.5%
(3) Treelet Kernel (TK)	208	205	209	212	99.1%
(4) TK with variable weighting	217	224	**223**	**250**	**99.7%**
(5) Graph Embedding	218	**227**	206	239	**99.7%**

the ten test sets for each class of animal for the first dataset and the accuracy obtained by differents methods on AIDS dataset. Note however that results obtained by tree pattern kernel are not displayed since the source code provided by the authors is restricted to molecules with a degree bounded by 4. First, we can note that method based on graph embedding (Table 2, Line 4) leads to globally better results than Gaussian kernel applied on an approximation of the graph edit distance (Table 2, Line 1). This observation highlights the better accuracy provided by the approximation of edit distance used in graph embedding method. In the same way, graph embedding methods outperforms Random Walks Kernel (Table 2, Line 2) and Treelet Kernel (Table 2, Line 4). However, combination of a variable weighting scheme with Treelet Kernel (Table 2, Line 4) improves the prediction accuracy of Treelet Kernel and obtains the best results on 3 over 5 datasets a slightly lower prediction accuracy than graph embedding methods on the two others. However, weighting each treelet according to a property to predict requires about 30 minutes for each train set of PTC dataset whereas computational time of graph embedding is performed in about 74 seconds for each PTC dataset. The accuracy provided by variable weighting can thus be obtained at the cost of an high computational time.

5 Conclusion

As shown in our experiments graph kernels and graph embedding methods provide close results in most of experiments. This last point is expected since as stressed in this paper both approaches are closely related. The main difference of both approaches should rather be determined from their potential usage. On one hand, Graph embedding methods provide an explicit embedding in a finite dimensional space for each input data sets. Hence, this approach is not restricted to kernel methods but can use explicitly the coordinates associated to graphs. On the other hand, this approach requires the whole data set to compute an embedding. Graph kernels based on bag of patterns, only require to compute the similarity between an input graph and the one of the training set. These methods may thus be used on unbounded data sets. The choice between both approaches should thus be determined from the ability for a given application to obtain the whole data set and from the ability of algorithms applied on graphs to be kernelized.

Acknowledgments

The authors thanks Salim Jouili for providing the graph embedding code.

References

1. D. Villemin B. Gaüzére, L. Brun and M. Mokhtari-Brun. Graph kernels based on relevant patterns and cycle information for chemoinformatics. In *Proceedings of ICPR 2012*, 2012. To be published.
2. H. Bunke. Error correcting graph matching: On the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):917–922, 1999.
3. J. Dattorro. *Convex Optimization and Euclidean Distance Geometry*. Meboo Publishing USA, 2005.
4. C. de Mauro, M. Diligenti, M. Gori, and M. Maggini. Similarity learning for graph-based image representations. *Pattern Recognition Letters*, 24(8):1115–1122, 2003.
5. T. Gärtner, P. A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and the 7th Kernel Workshop*, 2003.
6. Benoit Gaüzère, Luc Brun, and Didier Villemin. Two new graphs kernels in chemoinformatics. *Pattern Recognition Letters (In Press)*, (0):–, 2012.
7. S. Jouili and S. Tabbone. Graph matching based on node signatures. In *GbRPR*, pages 154–163, 2009.
8. S. Jouili and S. Tabbone. Graph embedding using constant shift embedding. In *ICPR Contests*, pages 83–92, 2010.
9. H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized Kernels Between Labeled Graphs. *Machine Learning*, 2003.
10. P. Mahé and J.-P. Vert. Graph kernels based on tree patterns for molecules. *Machine Learning*, 75(1):3–35, October 2009.
11. J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In *First International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74. Citeseer, 2003.
12. P. Ren, R. Wilson, and E. Hancock. Graph characteristics from the ihara zeta function. In da Vitoria Lobo et al., editor, *SSPR 2008*, volume 5342 of *Lecture Notes in Computer Science*, pages 257–266. Springer Berlin / Heidelberg, 2008.
13. K. Riesen and H. Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *SSPR 2008*, pages 287–297, Berlin, Heidelberg, 2008. Springer-Verlag.
14. K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vision Comput.*, 27(7):950–959, 2009.
15. V. Roth, J. Laub, M. Kawanabe, and J. M. Buhmann. Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(12):1540–1551, 2003.
16. Vishwanathan S.V.N., Schraudolph N., Kondor I. R., and Borgwardt. K. Graph kernels. *Journal of Machine Learning Research*, 11, April 2010.
17. H. Toivonen, A. Srinivasan, R. King, S. Kramer, and C. Helma. Statistical evaluation of the predictive toxicology challenge 2000-2001. *Bioinformatics*, 19(10):1183–1193, 2003.
18. A. Torsello and E. R. Hancock. Graph embedding using tree edit-union. *Pattern Recognition*, 40(5):1393–1405, 2007.